Hindawi
Scientific Programming
Volume 2022, Article ID 9607755, 15 pages
https://doi.org/10.1155/2022/9607755

Research Article

Construction of a Smart Supply Chain for Sand Factory Using the Edge-Computing-Based Deep Learning Algorithm

Bin Li,[1] **Xuejun Zhang**,[1,2,3] **Yanjiao Ban**,[1] **Xianfu Xu**,[1] **Wenjun Su**,[1] **Jingxian Chen**,[1] **Shan Zhang**,[1] **Feng Li**,[1] **Zuopeng Liang**,[3] **and Shengkai Zhou**[4]

[1]*School of Computer, Electronics and Information, Guangxi University, Nanning, China*
[2]*Guangxi Key Laboratory of Multimedia Communications and Network Technology, Guangxi, China*
[3]*Guangxi Xinhao Construction Engineering Co., Ltd, Guangxi, Nanning, China*
[4]*Yongkai Construction Group Co., Ltd, Guangxi, Nanning, China*

Correspondence should be addressed to Xuejun Zhang; xjzhang@gxu.edu.cn

Received 11 August 2022; Accepted 19 September 2022; Published 7 October 2022

Academic Editor: Lianhui Li

The diminishing natural sand has facilitated the booming of the sand manufacturing industry, and intelligent management of sand factories, in a time- and cost-efficient way, has become a growing tendency for the future. A role has been played in achieving intelligent management by constructing a smart supply chain. However, the smart sand factories are hardly involved in previously reported studies, which is inconsistent with related studies on smart factories and the Industrial Internet of Things (IIoT). In this paper, a smart supply chain management system (SSCMS) is constructed to realize the intelligence and automatization of the management of sand factories, using edge-computing and deep learning techniques. Along the supply chain, the deep learning model is used to realize the automatic identification of sand, avoiding the disadvantages of human identification, while improving the quality of sand factory operations. In order to relieve the pressure of network bandwidth, reduce system delay, and improve system operation efficiency, we use edge-computing technology to process data at the edge. To verify the performance of the constructed system, a sand factory simulation platform is established. Experiments show that the most critical indicator in the system, the accuracy rate of sand type identification, is above 98%, and the sand type identification time is only 0.022 s. In general, compared with traditional supply chain management, the constructed smart supply chain improves the quality and efficiency of sand factory operations, and all indicators of the designed system have achieved satisfactory results.

1. Introduction

Sand and aggregate are the largest, irreplaceable, and indispensable material for infrastructure construction, and it is closely related to the survival and development of human. Sand and aggregate account for 60% to 75% of the volume of concrete, and it is the main component of concrete. Therefore, human needs to consume tens of billions of sand and aggregate to build and transform the world every year. Sand commonly can be divided into natural sand and machine-made sand. Natural sand refers to naturally occurring sand, artificially mined with a particle size of less than 5 mm. Machine-made sand is the finished product processed through multiple processes using the sand-making machine. The characteristics of machine-made sand are that the finished product has good granularity. Moreover, various specifications of sand can be produced according to different material requirements during the manufacturing process. As the exploitation of natural sand resources continues, the natural sand resources that can be exploited are gradually decreasing. The reduction of exploitable natural sand resources has promoted the vigorous development of the sand-making industry, and therefore the number of sand-making factories has gradually increased.

The Internet of Things (IoT) is an intelligent network. It makes all devices connected to the network, and related communication is carried out through the network [1]. After the concept of IoT was proposed, scholars started to use IoT technology in different fields. Concepts such as smart medical healthcare [2, 3], smart buildings [4, 5], smart grid [6, 7], and smart transportation [8, 9] have begun to be proposed. The Industrial Internet of Things (IIoT) means the use of IoT technology in the industry, which makes industrial manufacturing intelligent [10, 11]. The development of IoT technology will undoubtedly make life smarter in the future.

With the improvement of computer computing power, deep learning has become popular. Deep learning is the process by which computers learn associations between known data and eventually process unknown data as intelligently as humans do [12]. Recently, more and more research has introduced deep learning technology to the IoT [13, 14]. Generally, the use of deep learning to process data requires computers with high-performance computing capabilities. Therefore, many researchers transfer data collected by edge devices to cloud servers with large computation power for processing [15]. However, this method has certain drawbacks. On the one hand, the increasing number of IoT devices will lead to an increasing amount of data being generated. If all the data generated by IoT devices at the edge were to be transmitted to cloud servers for processing, this would be a huge challenge for network bandwidth. On the other hand, the transmission of data will inevitably cause some delay, which is very unfriendly to applications that need to react in real-time.

Edge computing is defined as the processing of data on the side close to the source of generation [16]. The inclusion of edge computing in the IoT can solve the problem of network congestion caused by massive data transmission [17]. Edge computing becomes especially important when deep learning models are used in IoT devices to process data [18].

Field Programmable Gate Array (FPGA) is a typical parallel computing chip featured by the fast celerity, high stability, robust flexibility of computing properties. In contrast to the traditional algorithm, the computing speed of FPGA has increased by dozens, even hundreds of times while with power dissipation and economic cost declined comparatively [19]. FPGA has become popular in the field of deep learning due to its fast parallel computing characteristics [20]. Recently, it has become common to deploy FPGA in data centers [21], but it is an innovative research direction to use FPGA to accelerate programs in IoT. And there is no relevant research involved in deploying FPGA to the edge. But this is an exciting study because deploying FPGA to the edge can achieve more ideal real-time performance, which is good news for applications that require real-time response.

The supply chain is a functional network chain structure, which takes the enterprise as the core, uses raw materials to make final products, and finally delivers the products to consumers through the sales network [22]. Sand-making factories play the roles of suppliers, manufacturers, warehouses, and distribution centers in the supply chain. Building an intelligent supply chain is the key to achieving intelligent management of sand factory. At present, the management of the sand factory is still in the artificial management stage, and traditional supply chain technology is still used in the management process, which inevitably has the following problems:

(1) Most sand factories are still in the traditional mode of operation. Mechanical processing of sand, sand weighing, sand type identification, vehicle access management, and charging management all require a lot of human resources. Therefore, the labor cost of the sand factory accounts for a large proportion of the operating cost of the sand factory.

(2) Scattered distribution of sand resources and long distances between different sand factories. Managers need to spend a huge amount of time and effort to manage multiple sand factories.

(3) The sale of sand is the direct economic income source of the sand factory, and the unit price of different sand types is different. At present, when the sand factory sells the sand, the identification of the sand type is carried out by manual identification, which will inevitably cause problems such as errors in the identification of the sand type and corruption. These problems will directly lead to the loss of the benefits of the sand factory.

(4) The sand factory has a large number of large-heavy industrial vehicles and equipment operating in the sand factory. Therefore, there are potential mechanical hazards that may cause construction accidents. It will increase the risk of personnel disputes and accidents.

As Industry 4.0 evolves [23], IoT technologies are commonly used in the industry. And the proposed concept of Industry 4.0 has driven the development of smart supply chains [24]. Smart supply chain is an intelligent technology and management system built within the enterprise, integrating IoT technology with modern supply chain management theories, methods, and technologies [25]. How to construct the smart supply chain is a hot topic [26, 27], and the introduction of smart supply chain technology in sand factories can promote the realization of intelligent management of sand factories.

Aiming at the existing problems in the sand factory, we used edge-computing and deep learning technology to construct a smart supply chain management system (SSCMS) for the sand factory to realize the intelligence and automation of the sand factory management. In the system, we use the deep learning model to realize the automatic identification of sand types and use edge computing to deploy the model to the edge. At the edge, we have also innovatively added the FPGA to accelerate deep learning models. The use of edge servers and the FPGA can effectively alleviate network congestion, reduce communication delay, and decrease the burden on the cloud server. The overall performance of the system has been significantly improved.

The proposed work has the following contributions:

(1) Combining cloud computing, IoT, edge computing, deep learning, FPGA technology, and modern supply chain management theories and methods to

design a SSCMS for sand factories. The system can realize the intelligence and automation of sand factory management. It is specifically manifested in intelligent identification of sand loading vehicles, intelligent identification of sand types, automatic weighing of sand weight, cloud deduction of fees, and cloud management of sand factories.

(2) Identifying sand by deep learning approach relies on several dataset, but there are no publicly available relevant data sets. To achieve accurate sand identification, we collected a large number of pictures of different sand types and created our data set for sand identification.

(3) To deploy the sand recognition model to the edge, we propose an edge-based convolutional neural networks (CNN) model. Since the computation power of edge-computing nodes is far inferior to cloud servers, we optimized the parameters of the CNN model. The models deployed to the edge can reduce the amount of required computation while maintaining high recognition accuracy.

(4) We innovatively propose deploying FPGA at the edge to accelerate deep learning models. The use of FPGA reduces the processing delay of the system. And this study is innovative and there are no related studies on this subject.

(5) A microsand factory simulation platform is designed. The platform is used to simulate the operation process of a real-life sand factory. This simulation platform allows us to verify the performance of our designed SSCMS. We evaluate our method by experiments. The results show that our designed system realizes the intelligent management of the sand factory, and all indicators have achieved satisfactory results.

The rest of this paper is presented below. In Section 2, we describe in detail the basic architecture of the system being constructed. In Section 3, we present the design of the proposed edge-based CNN model for sand identification and analyze the model. In Section 4, we introduce the setup of the simulation platform and conduct experiments. Finally, we conclude in Section 5.

2. System Design

In this section, we mainly present the system architecture and workflow of the SSCMS. The architecture is mainly consisted of five parts, that is, sand factory gate system, license plate recognition system, weighing system, sand recognition system, and cloud platform management system, as shown in Figure 1.

Since sand resources are scattered, sand factories will have divisions in different locations. There are several IoT devices in each division. If the data generated by IoT devices in the factory of all divisions are transmitted to the cloud server for processing, the processing procedure will require a very large network bandwidth and cloud computation ability. The transmission of large amounts of data also impacts the transmission of control signals in the system.

Edge computing allows the processing of data on the side closer to data sources. We have deployed edge servers in each division of the sand factory, and the edge servers in different divisions process the data generated by IoT devices in their respective divisions. The edge server only needs to send the processed results back to the cloud server. Thus, it greatly reduces data transmission and improves system performance.

A complete sand purchase process is shown in Figure 2.

First, when the sand-purchasing vehicle enters the sand factory, the license plate recognition system activates the camera to capture the license plate picture and transmits the recognized license plate number as the user's ID to the edge server.

The gate system is opened, and then the sand-purchasing vehicle enters the sand identification weighing area, the system will count the weight of the empty vehicle and transmit the weight information back to the edge server. After obtaining the weight information, the edge server will associate the empty vehicle weight with the user information. After the weighing is completed, the sand-purchasing vehicle drives to the sand loading area. Then, the sand is loaded through the automatic device.

Then the sand identification weighing area is entered again, and at this time, the sand recognition system, weighing system, and license plate identification system will operate simultaneously. The sand recognition system identifies and uploads the type of sand purchased. The weighing system measures and uploads the weight information of the sand-purchasing vehicle. The license plate recognition system identifies and uploads the vehicle's identity information. The edge server will associate the uploaded sand type information, sand weight information, and user information.

Finally, by using the empty weight of the sand-purchasing vehicle, the loaded weight of the sand-purchasing vehicle and the type of sand, the cost of this sand purchase is calculated. If the user's balance of the account is sufficient, the edge server can automatically deduct the fee for the corresponding user. After the deduction is successful, the gate system opens and the sand-purchasing vehicle can leave the sand factory.

2.1. Cloud Platform Management System.

2.1. Cloud Platform Management System. The web interface of the cloud platform is developed by XOJO software. XOJO is a cross-platform programming language and visual development interface development tool. Compared with other tools, it supports cross-platform development and can develop multiplatform applications. Through cross-compilation, you can develop Linux system and Mac OS X system applications on Windows system. In addition, you can also develop multiple types of platform programs on the same system, such as desktop applications, network programs, console programs, and iOS mobile platforms. The web interfaces development of the cloud platform uses the development function of its network program. The cloud

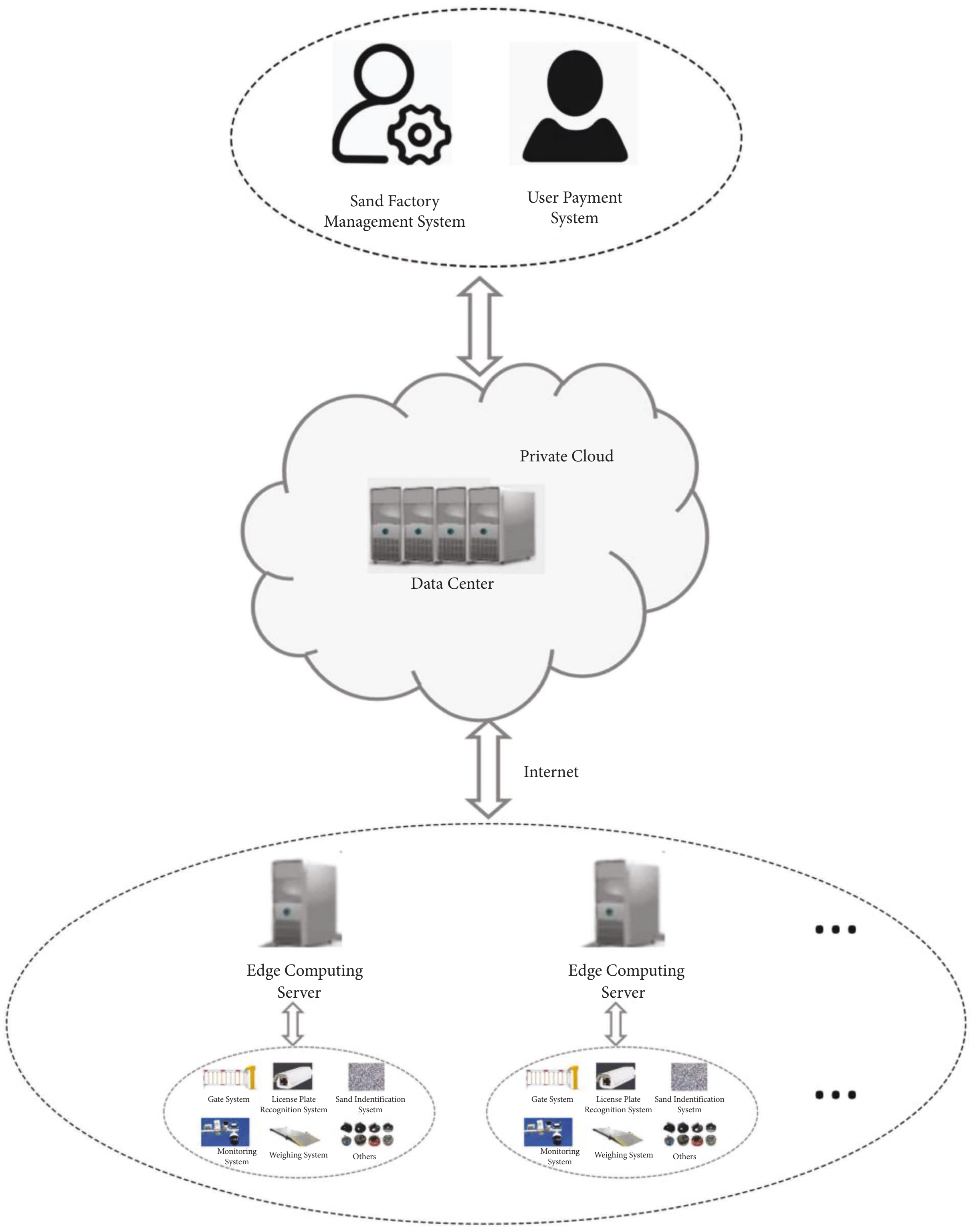


Figure 1: The system architecture of SSCMS.

platform is developed using the B/S architecture, with the client browser as the main application. This approach avoids the need for various applications on the client-side and concentrates on the core part of the system functionality implementation on the server. This approach simplifies the development, maintenance, and use of the system.

Only one browser is required to be installed on the client of this system. The edge server installs the MYSQL database and the executable program used on the edge side. They communicate with XOJO and are finally deployed on the web page and pushed to the client. This design makes the client application not restricted by the user's system

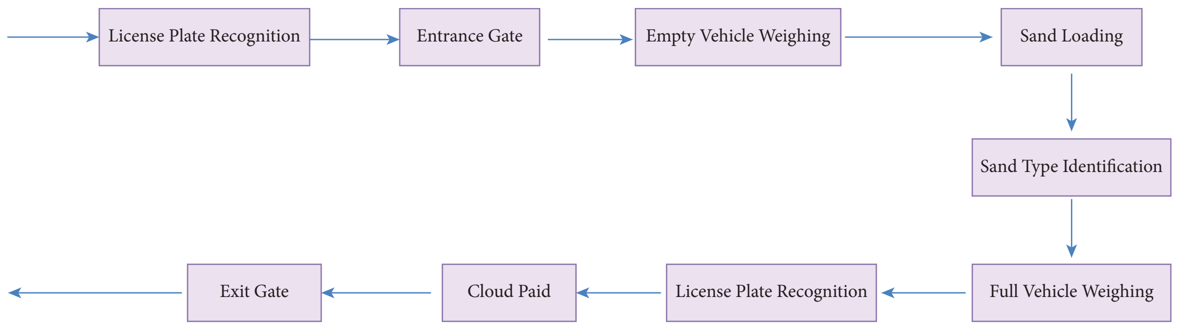


Figure 2: Sand purchase process.

platform and can be expanded on the mobile platform. Some algorithms with a relatively small amount of calculation and some database operation management can be implemented on the XOJO platform, but for algorithms with a relatively large amount of calculation, such as deep learning programs, it is necessary to use the program on the edge side to accelerate the network, that is, the program is placed in the server and it runs in the background. The cloud platform framework is shown in Figure 3.

Different from XOJO's network program, the edge-side program is not executed on the network channel but resides on the server-side in the form of an executable program group. The user sends a message through the browser, parses it through the XOJO web platform, and transmits it to the corresponding executable program. Once the program responds to this command, the client receives the corresponding result via the Internet. The platform is heterogeneity because the executable program on the edge side can be generated by any programming language, such as VS.net, Python, and MATLAB. At the same time, we designed to provide unified standard interface parameters to form an Algorithms Kernel module. Using this method, multiple developers can use different development tools to simultaneously develop programs in different programming languages, and the executable program group only needs to interact with XOJO in the end. This approach greatly shortens the development cycle of the entire cloud platform.

For some time-consuming algorithms, such as deep learning models, to ensure that the entire IoT system achieves an instant response, we use a combination of GPU + FPGA to accelerate the algorithm. FPGA is directly connected to the edge server and interacts with XOJO.

The cloud platform web interface includes the sand factory management interface and the user payment interface.

The web interface of sand factory management is designed to facilitate sand factory administrators to monitor the data of the sand factory at any time. It contains site operation, entering and exiting vehicles, the amount of sand sold on that day, the total amount of sand resources remaining on that day, the turnover of the sand factory, etc. The cloud platform web interface is shown in Figure 4 (the content in the colored area is for supplementary explanation). The administrator interface mainly includes four aspects: IoT system management, private cloud database management, user information management, and sand factory resource management. Among them, the IoT management system can manage all the IoT modules and also can view the operation of each module in real-time to achieve real-time monitoring. The database management systems can be used to check data in real-time. The user information management system can add or delete users and recharge the corresponding one. The sand factory resource management system can set the unit price of different sands, query the flow records of the sand factory, and export relevant data for verification.

The web interface of the user payment cloud platform is to provide sand factory clients with a convenient and fast way to purchase. The user payment cloud platform provides users with the option to reserve in advance the type of sand they need to purchase, the weight of the sand, and the time for arriving at the sand factory to load the sand. After the reservation is completed, the sand-purchasing vehicle can enter the sand factory for sand loading within the corresponding time. Moreover, users can also check their purchase records as well as their bill history.

2.2. License Plate Recognition System.

The role of the license plate recognition system is to obtain the identification information of the sand-purchasing vehicle. In a complete sand purchase process, the sand-purchasing vehicle needs to be identified twice. The first plate recognition is to record the vehicle information of the sand-purchasing vehicle when it enters the sand factory. A second plate recognition and a second vehicle weighing are carried out when the sand-purchasing vehicle is loaded with sand. The purpose of this license plate recognition is to correlate the sand-purchasing vehicle information, the sand purchase type, and the sand-purchasing vehicle load information.

License plate recognition system mainly adopts the technology of image processing and deep learning. The procedure can be divided into three steps, namely, image preprocessing, license plate detection, and license plate recognition. The first step is to preprocess the captured images, followed by license plate localization, and character segmentation. The license plate recognition system is directly deployed on the edge server-side with reference to existing research [28].

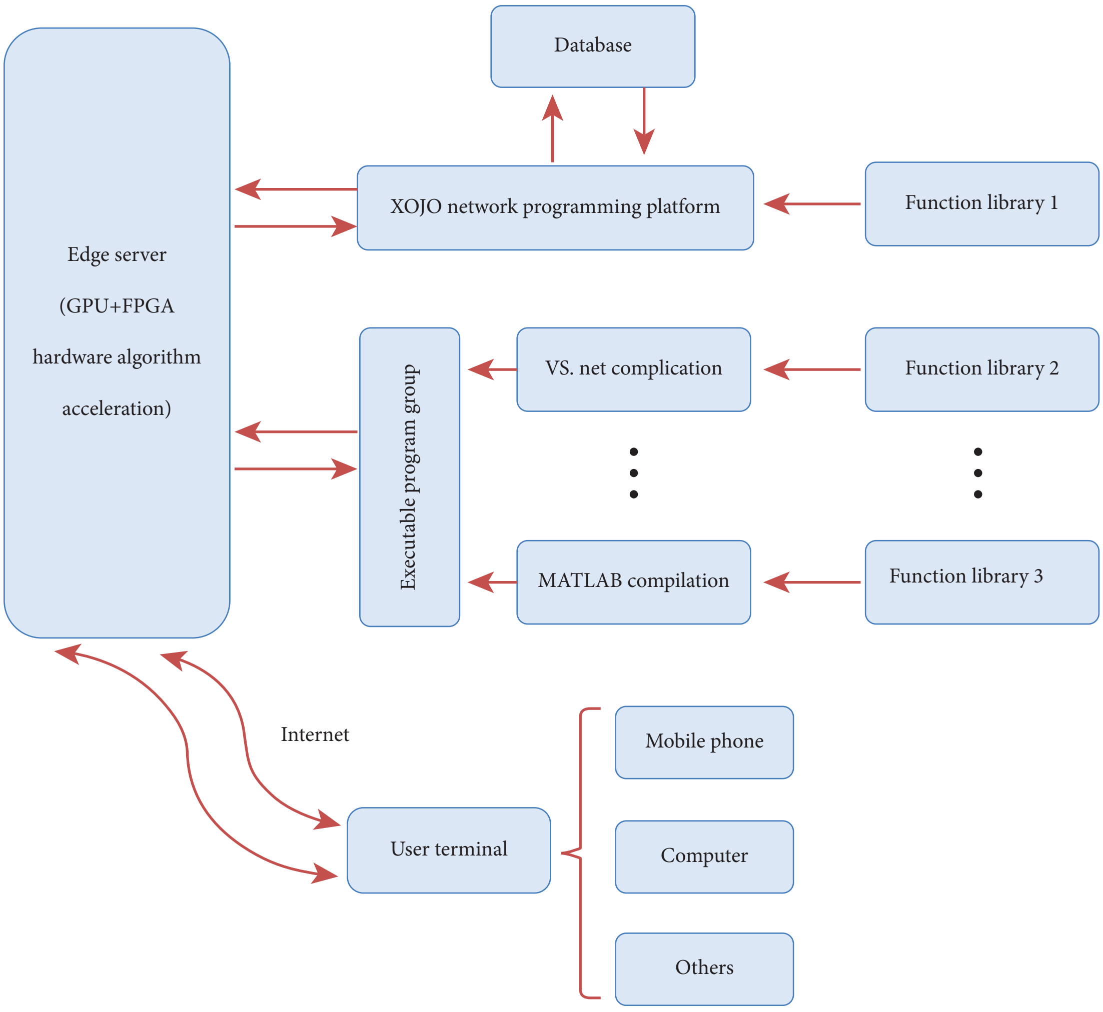


Figure 3: Cloud platform system framework.

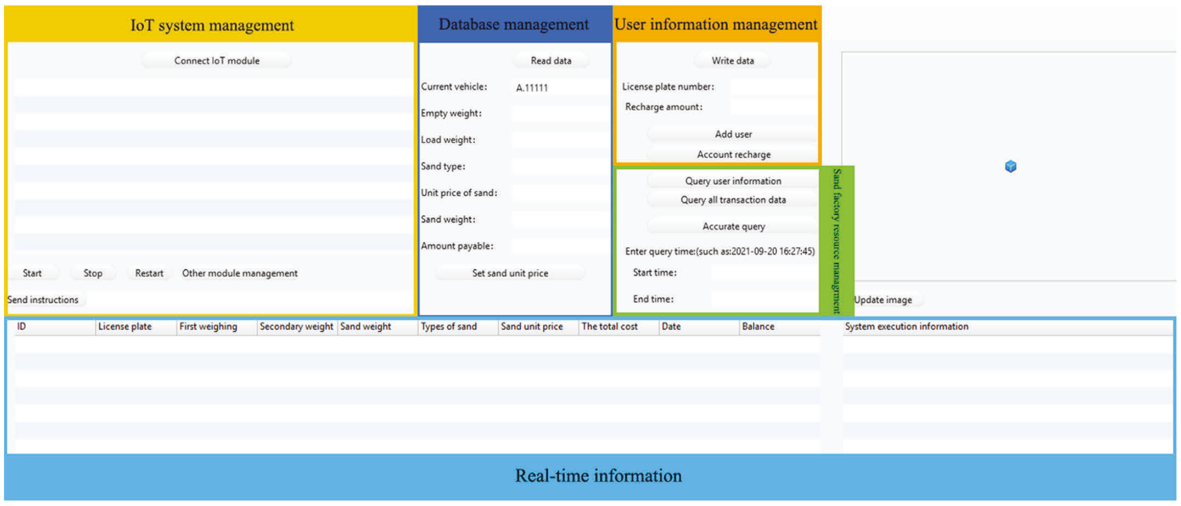


Figure 4: Web interface of sand factory management cloud platform.

2.3. Sand Weighing System. The sand weighing system mainly includes the weighbridge system and the IoT module. In a complete sand purchase process, the vehicle needs to be weighed twice. The weight when the vehicle is empty is recorded as W_0, the weight after loading with sand is recorded as W_1, and the weight of sand W is calculated as follows:

$$W = W_1 - W_0. \tag{1}$$

After the weighbridge system obtains the vehicle weight information, the relevant information needs to be transmitted to the edge server using the IoT module.

The unit prices of different sand types are saved in the cloud, and let the unit price of sand be calculated as P. The unit price of sand can be set on the server. Then the total cost of this sand purchase C is as follows:

$$C = P * W. \tag{2}$$

2.4. Sand Recognition System. Sand recognition system is to identify the type of sand. We use the CNN model to identify the type of sand. The identification of sand involves the processing of high-resolution images. If the images are transmitted to the cloud server for processing, it will be a severe challenge to the network bandwidth and will inevitably cause communication delays. The transmission of large amounts of data will affect the transmission of cloud server control signals, which will directly weaken the performance of the IoT system. Therefore, we propose an edge-based CNN model to identify the type of sand. Because the computing power of the edge server is far inferior to the cloud server, the parameters of the model are optimized. This enables us to reduce the amount of computation without losing recognition accuracy. And we put the recognition of sand in the FPGA at the edge to accelerate.

The workflow of the sand recognition system is as follows: the camera collects the sand pictures in the sand-purchasing vehicle, then transmits the pictures to the nearest edge server for picture preprocessing, and finally the pictures are preprocessed and transmitted to the FPGA for sand recognition. After the sand type is successfully identified, the FPGA will send the result back to the edge server.

2.5. Gate System. Gate system is to control the entry and exit of vehicles. For the sand-purchasing vehicles, before entering the sand factory, the vehicle information is recorded through the license plate recognition system. After that, the gate system opens the gate and releases the vehicle. When the sand-purchasing vehicle is loaded and the corresponding user's deduction is completed in the cloud, the gate system opens the gates to allow sand-purchasing vehicles to leave. At this point, the entire sand purchase process is completed. Besides, we can preload the corresponding license plate in the cloud and then automatically release the vehicles of the staff inside the factory.

3. CNN Model Design

This section mainly introduces the deep learning model design for sand recognition.

3.1. Data Collection and Preprocessing. The sand type recognition method based on deep learning has extremely high requirements for data sets. It is generally believed that the larger the dataset, the higher the recognition accuracy. However, there is no relevant available public sand dataset, so we collect sand pictures to make the dataset by ourselves.

At present, the sand factory mainly includes three types of sand. They are fine sand, coarse sand, and natural sand, respectively. Among them, natural sand is produced by natural forces, mainly rock particles, interspersed with a small amount of rock debris and mud debris sediment of rivers, lakes, and seas, and the grain size of rock particles is below 5 mm. Machine-made sand is the artificial mechanical crushing of hard and unweathered rock. The specifications of the machine-made sand are divided into three types: coarse, fine, and natural, according to the average particle size. The average particle size of coarse sand is 0.5 mm or more. The average particle size of fine sand is 0.25 mm to 0.35 mm. We collected pictures of three kinds of sand: fine sand, coarse sand, and natural sand. The representative pictures of different sands are shown in Figure 5. Figure 5(a) shows the fine sand, and the average particle size of the sand is 0.35 mm–0.25 mm, which is smaller than coarse sand and has less mud than natural sand. Figure 5(b) shows coarse sand, with an average particle size of 0.5 mm–0.35 mm. Compared with fine sand, the particle size of coarse sand is larger, and compared with natural sand, there is only a small amount of mud. Figure 5(c) shows the natural sand, the particle size is between fine sand and coarse sand and contains more mud.

The acquisition process is as follows: first, we spread the sand in the container. Second, we fixed the camera at 1.5 meters directly above the container. A different type of sand was used each time for the photographs. The final collection were 990 pictures for each type of sand.

The deep learning model used for sand recognition is deployed in the edge server, but the computation power of the edge server is far inferior to the cloud server. Therefore, we compressed the collected sand pictures to reduce the computational pressure on the edge server. Each compressed picture of sand is 256×256 pixels.

3.2. Optimal Convolutional Neural Network Structure. Deep Learning is currently the most popular machine learning method. CNN is the most common and stable technology for image classification and recognition. CNN enables images to be used as input to the network model and can automatically learn the training data. It overcomes the accumulation of errors generated by traditional manual feature extraction and achieves the extraction of higher-level features [29]. Combining the traditional CNN model with edge deployment, we propose an optimized CNN model suitable for sand classification.

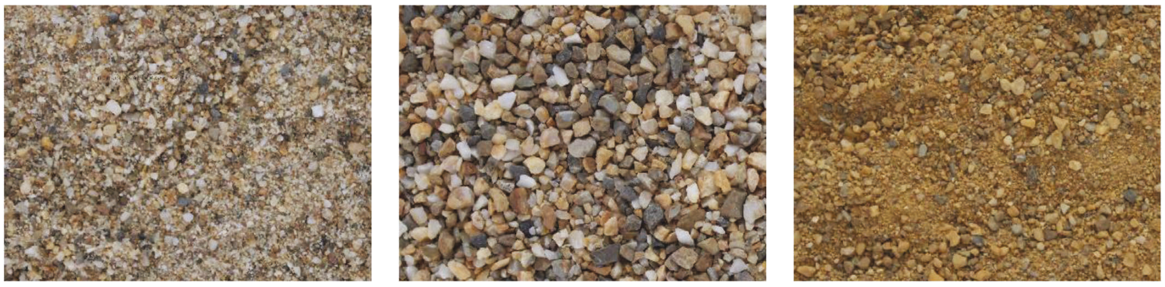


Figure 5: Representative images of different types of sand: (a) fine sand; (b) coarse sand; (c) natural sand.

The dataset for each type of sand is 990, for a total of 2970. In the initial experiment, we randomly divided all sampled data into training, validation, and test sets in the ratio of 0.7 : 0.15 : 0.15. We set the learning rate to 0.01. With the training method of momentum stochastic gradient descent, we set the momentum parameter to 0.9, MiniBatch to 64. First of all, the number of layers of convolutional layers needs to be determined by a quantitative experiment, as it is the most important parameter that directly affects the performance of the CNN model. Generally, convolutional layers (Conv) are used in combination with rectified linear unit (ReLU) layers and batch normalization (BN) layers. Therefore, we combine them. The performance of different Conv_ReLU_BN layers is shown in Figure 6.

Figure 6 shows the effect of the number of layers of Conv_ReLU_BN on the classification accuracy and training speed of the CNN model, where ч axis represents the number of Conv_ReLU_BN layers, and y axis represents the values of classification accuracy and training speed. From the figure, it can be obtained that when the number of layers of Conv_ReLU_BN is greater than 4, the accuracy no longer fluctuates substantially; when the number of layers of Conv_ReLU_BN increases, the training time increases. Therefore, we have chosen to keep 4 convolutional layers for feature extraction, and Table 1 shows the detailed parameters.

Secondly, the number of fully connected layers (FC) needs to be determined. FC is one of the key factors affecting the performance of the CNN model. Based on 4 Conv_ReLU_BN layers, we find the optimal number of FC layers by testing the effect of different numbers of FC layers on the CNN classification results. Figure 7 shows the effect of adding 1 to 5 FC layers on the classification results of the CNN model. In the figure, the ч axis represents the number of FC layers, and the y axis represents the value of accuracy and training speed. It can be seen from the figure that when the number of FC layers is greater than 2, the accuracy rate tends to be stable and less time-consuming at this point. Therefore, 2 FC layers were selected for classification, and the detailed parameters are shown in Table 2.

In addition, learning rate tuning is also an indispensable step to establishing a CNN model. The size of the learning rate will directly affect the convergence speed of the model. To make the 6-layer CNN model (4 Conv + 2 FC) converge faster with a higher accuracy rate, we choose different learning rates for experiments to find the best learning rate.

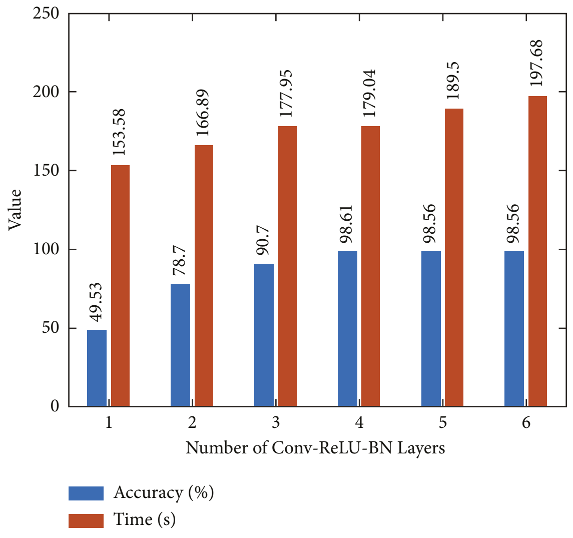


Figure 6: Optimal number of Conv_ReLU_BN layers.

Table 1: Feature extraction layers in proposed CNN structure.

Index	Layer name	Filter/pool size	No. of filters	Stride	Padding
1	Conv_ReLU_BN Maxpool	5 2	16	1 2	2 0
2	Conv_ReLU_BN Maxpool	5 2	32	1 2	2 0
3	Conv_ReLU_BN Maxpool	5 2	64	1 2	2 0
4	Conv_ReLU_BN Maxpool	5 2	128	1 2	2 0

Figure 8 shows the average result of 10 runs on the dataset. In the figure, the ч axis represents the learning rate, and the y axis represents the value of accuracy and training speed. As can be seen in Figure 8, the accuracy is maximum and the time consumed is relatively short when the learning rate is 0.01. Therefore, the optimal learning rate is determined to be 0.01.

Finally, in the experiment, we use the momentum stochastic gradient descent method. Thus, we need to determine the optimal MiniBatch. Other parameters were set as mentioned in the previous text, and we conducted ten experiments by changing the size of MiniBatch. Figure 9 shows

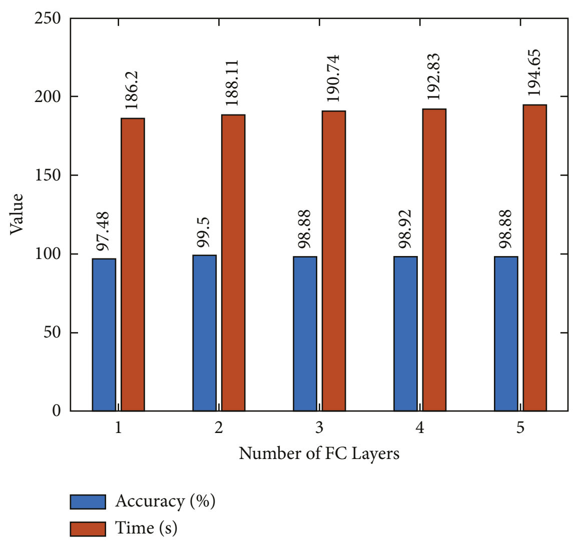


FIGURE 7: Optimal number of FC layers.

TABLE 2: Classification layers in proposed CNN structure.

Index	Layer name	Weights	Bias
5	FC_1	256×8192	256×1
6	FC_2	3×256	3×1
	Softmax		

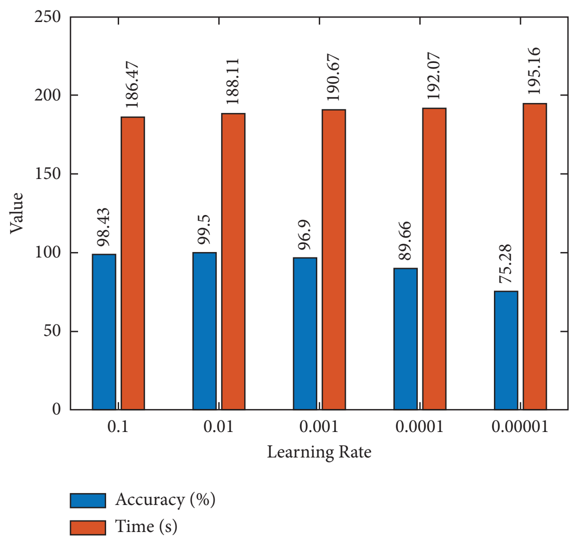


FIGURE 8: Optimal value of learning rate.

the average value after 10 experiments. In the figure, the ч axis represents the size of MiniBatch, and the y axis represents the value of accuracy and training speed. As we can observe from the figure, the highest accuracy and shorter time is achieved when MiniBatch is set to 64. Therefore, we determined the optimal MiniBatch size to be 64.

In summary, the optimal structure and hyperparameters of the CNN model are determined. The optimal structure is 4 Conv_ReLU_BN layers, 2 FC layers, and softmax layer; the learning rate and MiniBatch are 0.01 and 64, respectively.

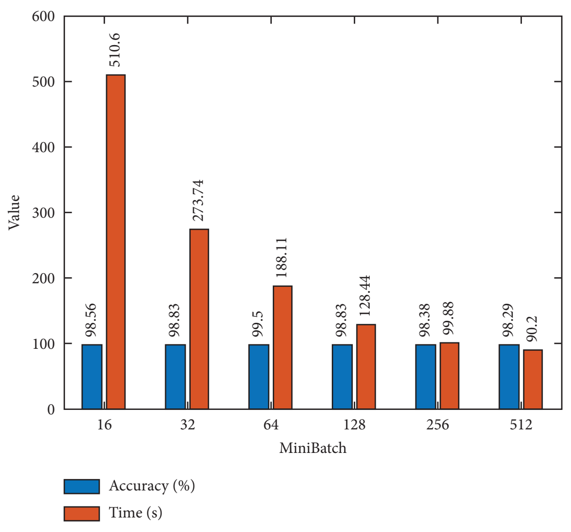


FIGURE 9: Optimal value of MiniBatch.

The training results of the optimal CNN model are shown in Figure 10.

As shown in Figure 10(a), the training accuracy of our CNN model increases with the number of iterations. Moreover, since 64 data are randomly selected at each update time, the curve fluctuates significantly before leveling off. In Figure 10(b), the training loss of the proposed CNN model decreases as the number of iterations increases and finally tends to zero. Figure 10 shows that the hyperparameters we have chosen are effective and they ensure the convergence of CNN.

3.3. Model Analysis. Generally, the number of convolutional layers will directly affect the model complexity. Then, the model complexity will directly affect the computational requirements in the model training process. And usually, the convolutional layer is followed deployed with the corresponding pooling layer. The total calculation time of the model can be calculated according to the number of convolutional layers and pooling layers, that is

$$T = \sum_{i=1}^{n} \left[N_c \left(L_{K_i}^2 + 1 \right) L_{I_i}^2 + \left(\frac{L_{I_{i+1}}}{L_{P_i}} \right)^2 \right] N_{F_i}, \tag{3}$$

where T represents the total calculation time, i represents the number of inputs, $N_c$represents the input image's number of channels, L_{K_i} represents the length of the convolution kernel, L_{I_i} represents the length of the input feature, L_{P_i}represents the size of the pooling layer, and N_{F_i}represents the number of convolution filters.

From (3), the time complexity of the model can be obtained as follows:

$$O\left(\sum_{i=1}^{n} L_{K_i}^2 L_{I_i}^2 N_{F_i} \right). \tag{4}$$

We compare the designed model with the popular LeNet-5 model [30] and the VGG-16 model [31]. The convolutional layers of the LeNet-5 model and VGG-16

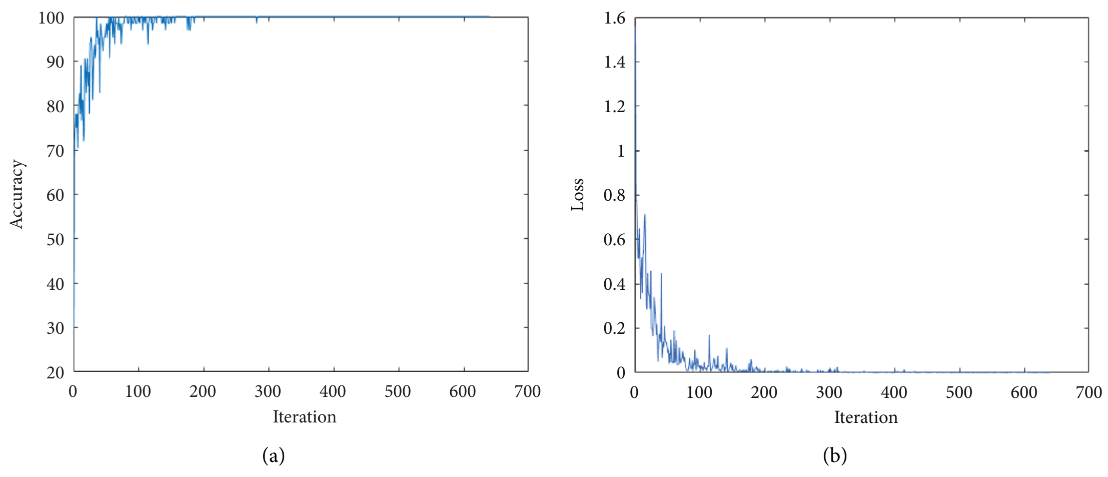


Figure 10: Training results of the optimal CNN model: (a) training accuracy; (b) training loss.

model are 6 and 16, respectively, while the model proposed in this paper uses only 4 convolutional layers. According to formula 3, the larger the number of convolutional layers, the greater the computational requirements of the model, so the model we propose requires the lowest computational requirements.

3.4. FPGA Design. After the model is determined, we use the FPGA to accelerate the designed deep learning model. The FPGA we used in the experiment was the Zynq UltraScale + MPSoC series board from Xilinx. The development process using the FPGA acceleration is shown in Figure 11.

After obtaining the model parameter file, we need to use Vitis AI Optimizer to trim the redundant connections in the neural network and reduce the overall required operations. Of course, this step is not necessary. After the model is simplified, the Vitis AI Quantizer tool needs to be used to quantify the model parameters, that is, floating-point to fixed-point. Vitis AI Quantizer can reduce computational complexity without loss of prediction accuracy. After the floating-point number is fixed-point, the network model requires less memory bandwidth, so it will provide faster speed and higher power efficiency than the floating-point model. Vitis AI Compiler compiles the quantized model into an efficient instruction set and data stream and can also perform complex optimizations, such as layer fusion, instruction scheduling, and reuse on-chip memory as much as possible. Vitis AI Profiler can conduct in-depth analysis on the efficiency and utilization of AI inference implementation. It can track function calls and running time and can also collect hardware information, including CPU, deep learning processing unit (DPU), and memory utilization. Vitis AI Library is a set of high-level libraries and APIs, which are built for efficient AI inference using the DPU. Vitis AI Library provides an easy-to-use and unified interface by encapsulating many efficient and high-quality neural networks. The Xilinx Runtime library enables applications to use a unified high-level running API at the edge, making edge deployments seamless and efficient.

4. Results

This section describes the relevant experiments and evaluates and analyzes the experimental results.

4.1. Experiment Platform. \We designed a microsand factory for simulating a real-life sand factory, as shown in Figure 12. In addition, we verified the functionality of the system using the designed SSCMS simulation platform.

In the simulation platform, we used the smart sand car to simulate the sand-purchasing vehicle in practical operation. As we discussed previously in Section 2, the red dotted areas in Figure 12 are the corresponding system modules for the simulation. In addition, all our subsystems use Wi-Fi wireless communication modules for communication with the server. The appearance of the wireless communication module used is shown in Figure 13.

4.2. System Verification. In this section, we perform functional tests of the SSCMS deployed in a microsand factory simulation platform. The configuration of the edge server is 2.5 GHz, 24-core CPU frequency, 32 G memory, and NVIDIA GeForce GTX 1650, with 8 T storage. Conducting the first step of the experiment, we connect the FPGA to the edge server, open the web interface of the sand factory management cloud platform on the edge server, run the executable program group, and connect the relevant IoT modules. When the smart sand car drives to the license plate recognition area, the license plate recognition system will capture the license plate information and verify the vehicle identity. The empty vehicle will be weighed after successful identity verification. The relevant data of the weighing will be associated with the license plate information on the server. At this time, the web interface of the sand factory management cloud platform is shown in Figure 14.

Next, after the smart sand car is loaded with sand, it will go to the sand identification weighing area to identify the type of sand. Then the edge server preprocesses the acquired

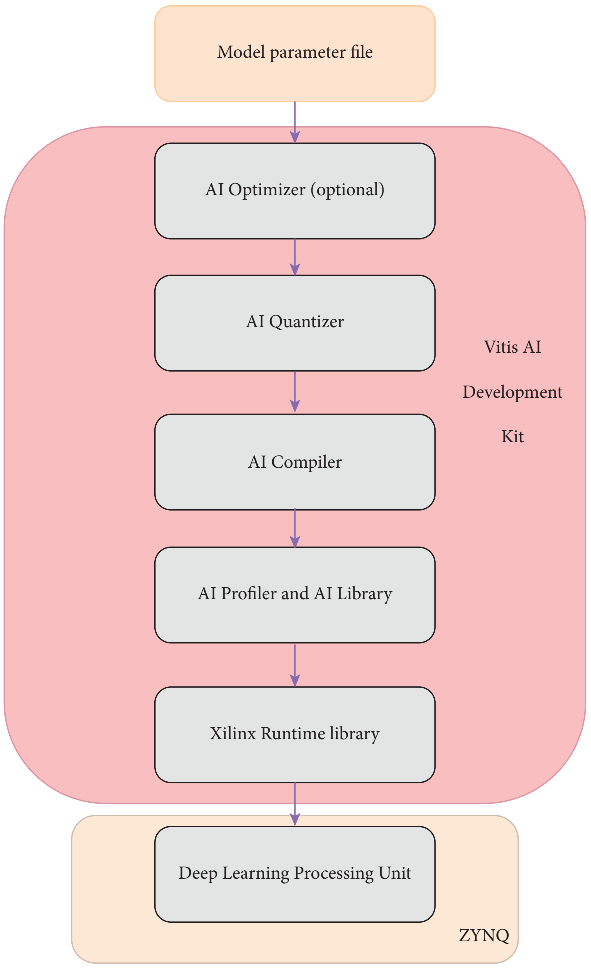


Figure 11: FPGA development process.

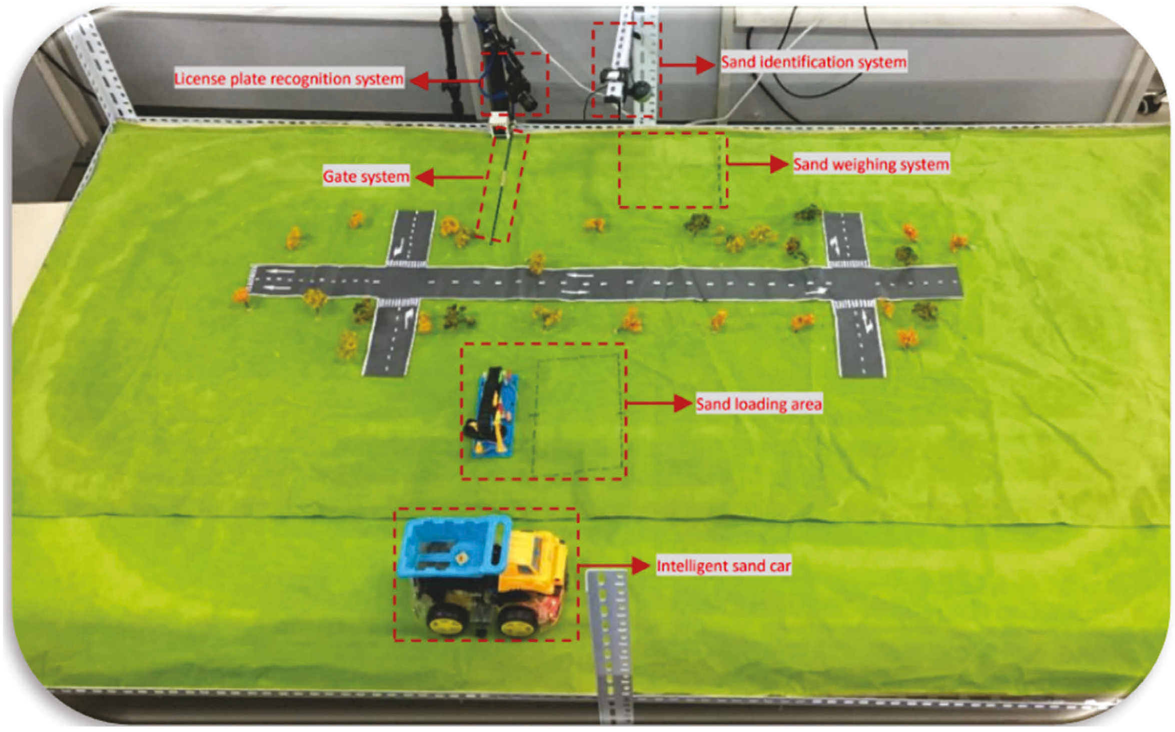


Figure 12: Microsand factory simulation platform.

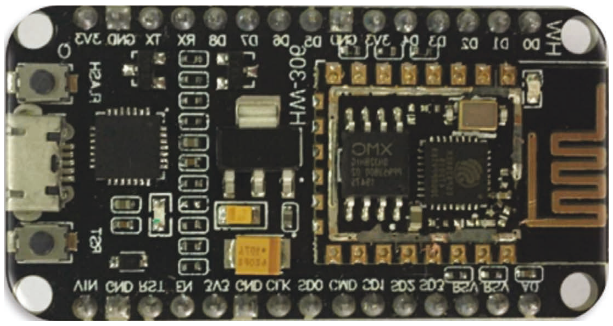
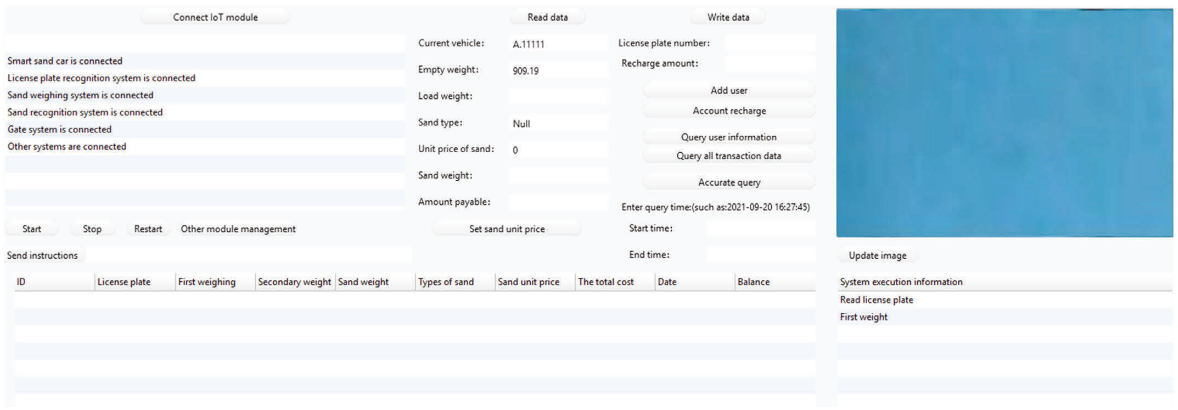

3V3 GND TX RX D8 D7 D6 D5 GND 3V3 D4 D3 D2 D1 D0
HW-30e
VIN GND RST EN 3V3 GND CLK SDO CMD SD1 SD2 SD3 RSV RSV A0

Figure 13: Wi-Fi wireless communication module.



Smart sand car is connected
License plate recognition system is connected
Sand weighing system is connected
Sand recognition system is connected
Gate system is connected
Other systems are connected

Current vehicle: A.11111
Empty weight: 909.19
Load weight:
Sand type: Null
Unit price of sand: 0
Sand weight:
Amount payable:

License plate number:
Recharge amount:
Enter query time:(such as:2021-09-20 16:27:45)
Start time:
End time:

Send instructions

ID	License plate	First weighing	Secondary weight	Sand weight	Types of sand	Sand unit price	The total cost	Date	Balance

System execution information
Read license plate
First weight

Figure 14: The web interface of empty vehicle status.

sand image, and the preprocessed image will be transferred to the FPGA, and the FPGA will use the designed CNN model to accelerate the identification of the sand type. At this time, the edge server will recognize the license plate and weigh the sand. Finally, vehicle information, sand type, and sand weight will be associated on the edge server-side. The unit price of sand can be set on the edge server-side, and different unit prices can be set for edge servers located in different locations. Using (1) and (2), we calculate the cost of this purchase and display it in the web-side interface of the cloud platform.

We set the unit price for three different types of sand. The unit prices of coarse sand, fine sand, and natural sand were set in the sever to 80, 100, 70, respectively. The operating results are shown in Figures 15–17 respectively. The edge server will calculate the cost of this purchase and then deduct the fee for the corresponding user. After the fee is successfully deducted, the gate will open to let the smart sand car drive out.

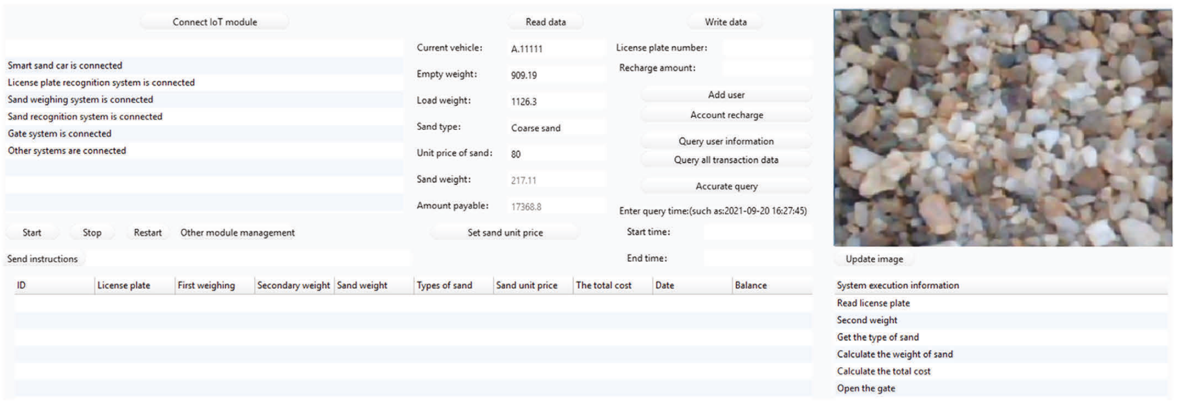
In the web-side interface of the sand factory management cloud platform, we can access the user's identity information. Including license plate number, the weight of empty vehicle, weight of loaded vehicle, the weight of sand, type of sand, and the cost of this purchasing. We can obtain a picture of the captured sand from the right of the interface. From the figure, the system can accurately identify different types of sand and their weight information. Eventually, the system will calculate the total cost accurately based on the set unit price of sand.

To test the performance of the built system, we run the system 100 times. Moreover, we have conducted statistic accuracy on sand recognition, license plate recognition, and sand weighing. In these 100 experiments, the type of sand, the license plate number, and the weight of the sand were changed for each experiment. After 100 experiments, the results are shown in Table 3.

In Table 3, the accuracy rate of sand recognition, license plate recognition, and sand weighing of our system is all up to 98 and above.

In our paper, the use of edge servers makes it possible to process data in each division, and the use of the FPGA relieves the computing pressure of the edge server and reduces the system latency. After the data is processed at the edge, it is only necessary to transmit the processing results back to the cloud server. The sand recognition system and the license plate recognition system use deep learning models. The time consumed by model inference is a key metric for evaluating a system. The use of edge servers can significantly reduce data transfer, thereby reducing the time consumed by data transfer. Therefore, we deploy the models to the edge server to avoid unnecessary image transmission. Similar to the recognition accuracy experiments in the previous section, we run the deep learning model system deployed to the edge 100 times. We count the running time of the sand recognition and license plate recognition system. The average value of the results are shown in Table 4.

Table 4 illustrates the very short recognition time of this deep learning model deployed to the edge. Compared to the model deployed to the cloud server, this method reduces the



Smart sand car is connected
License plate recognition system is connected
Sand weighing system is connected
Sand recognition system is connected
Gate system is connected
Other systems are connected

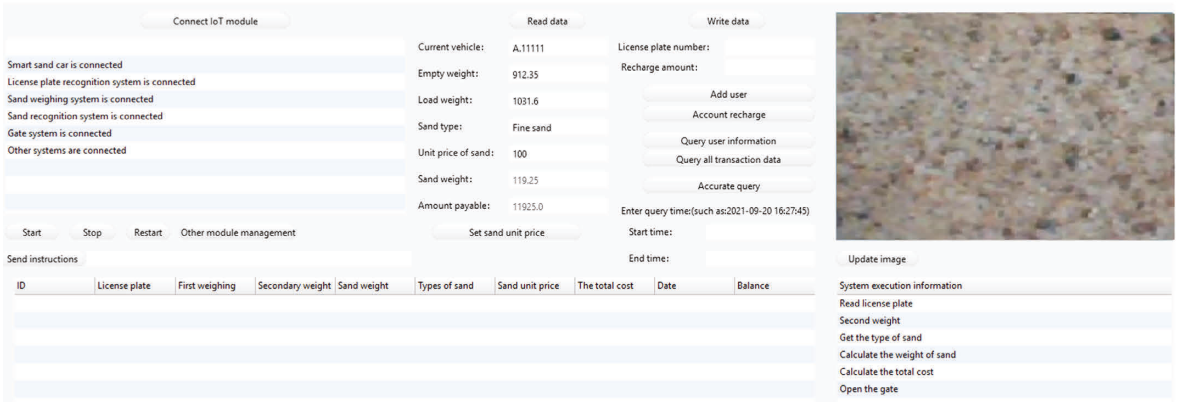
Current vehicle: A.11111
Empty weight: 909.19
Load weight: 1126.3
Sand type: Coarse sand
Unit price of sand: 80
Sand weight: 217.11
Amount payable: 17368.8

License plate number:
Recharge amount:
Enter query time:(such as:2021-09-20 16:27:45)
Start time:
End time:
Send instructions

ID	License plate	First weighing	Secondary weight	Sand weight	Types of sand	Sand unit price	The total cost	Date	Balance

System execution information
Read license plate
Second weight
Get the type of sand
Calculate the weight of sand
Calculate the total cost
Open the gate

Figure 15: Web interface for coarse sand recognition.



Smart sand car is connected
License plate recognition system is connected
Sand weighing system is connected
Sand recognition system is connected
Gate system is connected
Other systems are connected

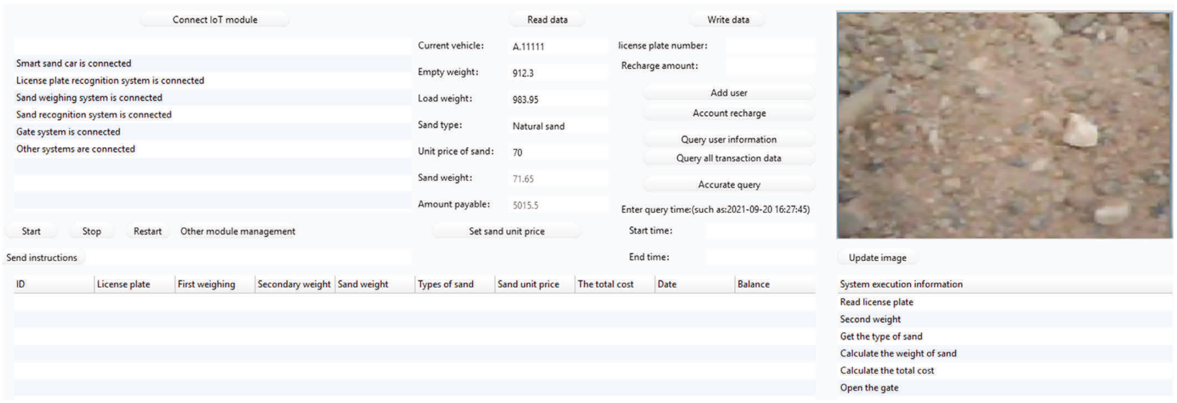
Current vehicle: A.11111
Empty weight: 912.35
Load weight: 1031.6
Sand type: Fine sand
Unit price of sand: 100
Sand weight: 119.25
Amount payable: 11925.0

License plate number:
Recharge amount:
Enter query time:(such as:2021-09-20 16:27:45)
Start time:
End time:
Send instructions

ID	License plate	First weighing	Secondary weight	Sand weight	Types of sand	Sand unit price	The total cost	Date	Balance

System execution information
Read license plate
Second weight
Get the type of sand
Calculate the weight of sand
Calculate the total cost
Open the gate

Figure 16: Web interface of fine sand recognition.



Smart sand car is connected
License plate recognition system is connected
Sand weighing system is connected
Sand recognition system is connected
Gate system is connected
Other systems are connected

Current vehicle: A.11111
Empty weight: 912.3
Load weight: 983.95
Sand type: Natural sand
Unit price of sand: 70
Sand weight: 71.65
Amount payable: 5015.5

license plate number:
Recharge amount:
Enter query time:(such as:2021-09-20 16:27:45)
Start time:
End time:
Send instructions

ID	License plate	First weighing	Secondary weight	Sand weight	Types of sand	Sand unit price	The total cost	Date	Balance

System execution information
Read license plate
Second weight
Get the type of sand
Calculate the weight of sand
Calculate the total cost
Open the gate

Figure 17: Web interface of natural sand recognition.

Table 3: System recognition accuracy.

Function	Accuracy (%)
Sand recognition	98
License plate ecognition	98
Sand weight	99.3

Table 4: System recognition time.

Function	Time (s)
Sand recognition	0.022
License plate recognition	0.073

time consumed by the image transfer. The recognition of sand is accelerated in the FPGA, and the license plate recognition is performed in the edge server. In the table, the sand type recognition time is significantly lower than the license plate recognition time because we use the edge server combined with FPGA to accelerate the sand type recognition, while the license plate recognition only uses the edge server for inference. The experimental results show that the use of FPGA can greatly reduce the computing pressure of edge servers and improve computing performance.

In summary, the SSCMS we have constructed has enabled intelligent and automated management of the sand factory, with satisfactory results in all indicators. Different from the cloud server–based model, our designed edge-based model avoids the time consumed by a large amount of data transmission, thus improving the performance of the entire system. Compared with traditional supply chain management, the constructed smart supply chain improves the operation quality and efficiency of the sand factory. Compared with the traditional supply chain, the constructed smart supply chain has the following advantages: stronger technology penetration, stronger visualization, stronger information integration, stronger extensibility, stronger collaboration, and stronger agility.

5. Conclusion

This paper addresses the drawbacks of the traditional supply chain and constructs a SSCMS to realize the intelligent management of sand factories. It is specifically manifested in intelligent identification of sand loading vehicles, intelligent identification of sand types, automatic weighing of sand weight, cloud deduction of fees, and cloud management of sand factories. Along the supply chain, we build the optimal deep learning model, which achieves automatic sand identification, avoiding the drawbacks of manual identification and improving the quality of sand factory operations. In addition, considering that edge-computing technology can reduce the transmission of a large amount of data, thus serving to relieve the pressure on network bandwidth and reduce system latency, we make full use of the superiority of edge-computing technology to effectively improve the operational efficiency of the sand factory. A sand factory simulation platform was established to verify the performance of the system. Experiments show that all indicators of the designed system have achieved satisfactory results. In general, compared with traditional supply chain management, the constructed smart supply chain improves the quality and efficiency of sand factory operations.

Recently, we have also deployed the designed system to the actual sand factory for testing. In reality, the applicability of our model has also been verified and we achieved a satisfactory result both in accuracy and speed. In the next research, we will make more efforts in promoting the practical application of the system and continue to improve the system for new problems.

Data Availability

The data are derived from Guangxi Xinhao Construction Engineering Co., Ltd, Nanning, Guangxi, China, and Yongkai Construction Group Co., Ltd, Nanning, Guangxi, China.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the Science and Technology Key Projects of Guangxi Province (Grants no. 2020AA21077007 and Z20210069) and the Innovation Project of Guangxi Graduate Education YCSW2022042.

References

[1] D. Silva, A. Heideker, I. D. Zyrianoff et al., "A management architecture for IoT smart solutions: design and implementation," *J NetwSyst Manage*, vol. 30, no. 2, p. 35, 2022.

[2] T. Lo'ai, M. Fadi, T. Mais, Q. Muhannad, and A. A. E.-L. Ahmed, "Edge enabled IoT system model for secure healthcare," *Measurement*, vol. 192, Article ID 110792, 2022.

[3] H. B. Mahajan, A. S. Rashid, A. A. Junnarkar et al., "Integration of healthcare 4.0 and blockchain into secure cloud-based electronic health records systems," *ApplNanosci*, Springer, Berlin/Heidelberg, Germany, 2022.

[4] A. A. Ahmed, A. C. Mihaela, A. J. Brimicombe, S. A. Ghorashi, A. Baravalle, and P. Falcarin, "Data quality challenges in large-scale cyber-physical systems: A systematic review," *Information Systems*, vol. 105, Article ID 101951, 2022.

[5] W. Zhang, W. Z. Hu, and Y. G. Wen, "Thermal comfort modeling for smart buildings: a fine-grained deep learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2540–2549, 2019.

[6] J. Gao, X. G. Wang, and W. M. Yang, "SPSO-DBN based compensation algorithm for lackness of electric energy metering in micro-grid," *Alexandria Engineering Journal*, vol. 61, no. 6, pp. 4585–4594, 2022.

[7] H. Wang, X. Wen, Y. Xu, B. Zhou, J. Peng, and W. Liu, "Operating state reconstruction in cyber physical smart grid for automatic attack filtering," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 2909–2922, 2022.

[8] C. Celes, A. Boukerche, and A. A. F. Loureiro, "Mobility trace analysis for intelligent vehicular networks: Methods, models, and applications," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–38, 2022.

[9] T. Baker, M. Asim, H. Samwini, N. Shamim, M. M. Alani, and R. Buyya, "A blockchain-based Fog-oriented lightweight framework for smart public vehicular transportation systems," *Computer Networks*, vol. 203, Article ID 108676, 2022.

[10] F. Zhou, L. Feng, M. Kadoch, P. Yu, W. Li, and Z. Wang, "Multiagent RL aided task offloading and resource management in wi-fi 6 and 5G coexisting industrial wireless environment," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 2923–2933, 2022.

[11] T. Li, Y. Ma, K. Yoshikawa, O. Nomura, and T. Endoh, "Energy-efficient convolution module with flexible bit-adjustment method and ADC multiplier architecture for industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3055–3065, 2022.

[12] C. W. Tian, L. K. Fei, W. Zheng, Y. Xu, W. Zuo, and C. W. Lin, "Deep learning on image denoising: an overview," *Neural Networks*, vol. 131, pp. 251–275, 2020.

[13] M. Esmail Karar, A. H. Abdel-Aty, F. Algarni, M. Fadzil Hassan, M. Abdou, and O. Reyad, "Smart IoT-based system for detecting RPW larvae in date palms using mixed depthwise convolutional networks," *Alexandria Engineering Journal*, vol. 61, no. 7, pp. 5309–5319, 2022.

[14] Q. Liu, T. Xia, L. Cheng, M. van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in IoT edge systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1491–1502, 2022.

[15] N. Krishnaraj, M. Elhoseny, M. Thenmozhi, M. M. Selim, and K. Shankar, "Deep learning model for real-time image compression in Internet of Underwater Things (IoUT),"

Journal of Real-Time Image Processing, vol. 17, no. 6, pp. 2097–2111, 2020.

[16] X. Wang, Z. Ning, L. Guo, S. Guo, X. Gao, and G. Wang, "Online learning for distributed computation offloading in wireless powered mobile edge computing networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1841–1855, 2022.

[17] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of Things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[18] H. Li, K. Ota, and M. X. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[19] L. Bustio-Martínez, R. Cumplido, M. Letras, R. Hernandez-Leon, C. Feregrino-Uribe, and J. Hernandez-Palancar, "FPGA/GPU-based acceleration for frequent itemsets mining: A comprehensive review," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–35, 2022.

[20] H. Cho, J. Lee, and J. Lee, "FARNN: FPGA-GPU hybrid acceleration platform for recurrent neural networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1725–1738, 2022.

[21] Z. Zhu, A. X. Liu, F. Zhang, and F. Chen, "FPGA resource pooling in cloud computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 610–626, 2021.

[22] X. L. Wang, Y. Y. Qiu, J. Chen, and X. Hu, "Evaluating natural gas supply security in China: An exhaustible resource market equilibrium model," *Resources Policy*, vol. 76, Article ID 102562, 2022.

[23] F. Siqueira and J. G. Davis, "Service computing for industry 4.0: State of the art, challenges, and research opportunities," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–38, 2022.

[24] N. Kazantsev, G. Pishchulov, N. Mehandjiev, P. Sampaio, and J. Zolkiewski, "Investigating barriers to demand-driven SME collaboration in low-volume high-variability manufacturing," *Supply Chain Management: International Journal*, vol. 27, no. 2, pp. 265–282, 2022.

[25] S. K. Sardar, B. Sarkar, and B. Kim, "Integrating machine learning, radio frequency identification, and consignment policy for reducing unreliability in smart supply chain management," *Processes*, vol. 9, no. 2, p. 247, 2021.

[26] N. Azizi, H. Malekzadeh, P. Akhavan, O. Haass, S. Saremi, and S. Mirjalili, "IoT-Blockchain: Harnessing the power of Internet of thing and blockchain for smart supply chain," *Sensors*, vol. 21, no. 18, p. 6048, 2021.

[27] S. E. Chen, S. Brahma, J. Mackay, C. Cao, and B. Aliakbarian, "The role of smart packaging system in food supply chain," *Journal of Food Science*, vol. 85, no. 3, pp. 517–525, 2020.

[28] H. Wang, Y. Li, L. M. Dang, and H. Moon, "Robust Korean license plate recognition based on deep neural networks," *Sensors*, vol. 21, no. 12, p. 4140, 2021.

[29] X. L. Zhang, M. L. Shen, X. Li, and F. Feng, "A deformable CNN-based triplet model for fine-grained sketch-based image retrieval," *Pattern Recognition*, vol. 125, Article ID 108508, 2022.

[30] W. Imen, M. Amna, B. Fatma, S. Fatma Ezahra, and N. Masmoudi, "Fast Hevc intra-CU decision partition algorithm with modified Lenet-5 and alexnet," *Signal, Image and Video Processing*, vol. 16, 2022.

[31] D. Tiwari, M. Dixit, and K. Gupta, "Deep multi-view breast cancer detection: A multi-view concatenated infrared thermal images based breast cancer detection system using deep transfer learning," *Traitement du Signal*, vol. 38, no. 6, pp. 1699–1711, 2021.